Theoretical Computer Science 87 (1991) 329-346
Elsevier

Machine characterization of (E0L-E0L) array languages

Nalinakshi Nirmal *

Department of Mathematics, Madras Christian College, Madras 600 059, India

R. Rama

Department of Mathematics, Madras Institute of Technology, Madras 600 044, India

Communicated by M. Nivat
Received March 1989

Abstract

Nirmal, N. and R. Rama, Machine characterization of (E0L-E0L) array languages, Theoretical Computer Science 87 (1991) 329-346.

A traditional topic in formal language theory, is to characterize classes of languages by machine models. Extending the idea of machine characterization of L systems to two dimensions and at the same time to generate pictures which are not rectangular, we propose in this paper a new model called (E0L-E0L) array system. (E0L-E0L) array languages are obtained by substituting E0L languages vertically into E0L languages. First a horizontal line of intermediates is generated by an E0L system. Then E0L languages are substituted vertically for each intermediate cell resulting in a two-dimensional language which need not be rectangular. We propose in this paper (Restricted Pushdown Array of Counters-Restricted Pushdown Array of Counters) array automata ((RPAC-RPAC)AA) and we show that a language is a (RPAC-RPAC)AA language if and only if it is a (E0L-E0L)A language.

Introduction

In the last few years, the study of developmental systems or L systems has become a very actively investigated topic of research [1, 11]. Recently many investigators have attempted to incorporate the developmental type of generation used in L systems to higher dimensions [2, 3, 4, 7, 9, 13]. In [2], a two dimensional array model called 0L and T0L array systems is introduced where parallel rewriting of every symbol in a rectangular array is considered and at each step of derivation

* Present affiliation: Bishop Cotton Women's Christian College (Karnataka Central Diocese) C.S.I., Field Marshall Cariappa Road (Residency Road), Bangalore 560 01, India.

every symbol is replaced by an array of the same dimension and the growth is exponential. In [4], Nirmal and Krithivasan introduced a new developmental model which expands linearly and whose rewriting rules are simple, elegant and free from parentheses and which generate interesting pictures like token I, T, H, L, kolam patterns, geometric patterns, etc.

At present, the mathematical theory of multi-dimensional L systems is much poorer than the theory of L systems generating sets of strings. To strengthen the mathematical theory of multi-dimensional L systems we study the effect of context dependent rules in table matrix L systems in [5], normal forms of extended table matrix L systems in [6] and finite index on ET0L array systems in [8].

None of the above mentioned models have a machine characterization. A topic of interest in formal language theory is to characterize classes of languages by machine models, as it helps us for intuitive reasoning about various properties of languages in a given class and it also plays a main role in the consideration of complexity of recognition and parsing of different language classes. Obtaining a machine characterization of L systems was an open problem for a long time. In L systems, the absence of the sequential way of re-writing is the main difficulty in finding a machine model. Due to this difficulty as the derivation proceeds, larger and larger number of occurrences of letters must be rewritten at a single step of derivation. Rozenberg in [10] introduced a machine model which takes care of this parallel rewriting process and with a certain restriction accepts the class of E0L languages.

Extending the idea of machine characterization of L systems to two dimensions, we propose in [7], an E0L-regular matrix system obtained by substituting regular sets into well-known families of L systems. We know that substitution is a very well-defined operator in formal language theory [12]. In this E0L-regular matrix system, the substitution operator operates vertically on L systems to yield matrix languages which are rectangular and we construct a Restricted Pushdown Array of Counters-finite matrix automaton which accepts E0L-regular matrix languages.

Motivated by the idea of obtaining nonrectangular arrays and to characterize it by a machine, we propose in this paper a new model (E0L-E0L) array language which is obtained by substituting E0L languages vertically into E0L languages. Intuitively, first a horizontal line of intermediate cells is generated by E0L (T0L, 0L) systems. Then into each intermediate cell in the horizontal line, a vertical E0L, T0L or 0L language is substituted resulting in a two-dimensional language which need not be rectangular. We observe that we can generate interesting pictures like token T, and some complex kolams. In picture processing, parallel generation or parallel recognition is very important as it reduces computer time in contrast to sequential processing which requires less hardware but more time consuming. We propose in this paper a parallel cum parallel machine called (Restricted Pushdown Array of Counters-Restricted Pushdown Array of Counters) array automaton which accepts (E0L-E0L) array languages. Whenever a new model is introduced, it is but natural to investigate its closure properties under various operators.

We investigate the closure properties of the nine families of languages obtained in this paper under some of the AFM operators like union, column catenation, column +, column * and picture language operators like, the reflection about the right most vertical [13]. We introduce three new operators: row cross doubling, column cross doubling, cyclic permutation and we also introduce column homomorphism and study the closure properties of the language families introduced in this paper. We compare these families of languages with some of the already existing two-dimensional language families of [2, 3, 4, 13].

This paper is divided into three sections. Section 1 deals with the definitions and examples. Section 2 deals with the machine characterization of (EOL-EOL) array languages. Section 3 deals with the closure properties of the language families of this paper under some of the AFM operators, picture language operators [13] and under column cross doubling, row cross doubling, cyclic permutation and column homomorphism.

1. Definitions and examples

In this section we define various two-dimensional developmental systems generating different families of array languages. For the definitions of 0L, T0L and E0L systems the reader is referred to [11].

Definition 1.1. An (E0L-E0L) array systems ((E0L-E0L)AS) is a two-tuple $G = (G_1, G_2)$, where $G_1 = (V_1 \cup I_1, I_1, P_1, S_0)$ is a E0L system with V_1 is a finite set of horizontal auxiliary symbols; $I_1 = \{S_1, \ldots, S_k\}$, a finite set of intermediates; P_1 is a finite nonempty subset of $(V_1 \cup I_1) \times (V_1 \cup I_1)^*$ called a set of horizontal productions satisfying the completeness condition that for each a in $V_1 \cup I_1$, there exists an α in $(V_1 \cup I_1)^*$ such that (a, α) denoted by $a \to \alpha$ belongs to P_1; $S_0 \in V_1$ is the start symbol or axiom and $V_1 \cap I_1 = \emptyset$.

$G_2 = \bigcup_{i=1}^{k} G_{2i}$ where $G_{2i} = (V_{2i} \cup I_{2i}, I_{2i}, P_{2i}, S_i)$, $i = 1, \ldots, k$ are k E0L systems with $I_2 = \bigcup_{i=1}^{k} I_{2i}$ is a finite nonempty set of terminals, $I_{2i} \cap I_{2j}$ need not be empty for $i \neq j$: V_{2i} is a finite nonempty set of vertical auxiliary symbols such that $V_{2i} \cap V_{2j} = \emptyset$ for $i \neq j$; P_{2i} is a finite nonempty set of $(V_{2i} \cup I_{2i}) \times (V_{2i} \cup I_{2i})^*$ satisfying the completeness condition that for each a in $(V_{2i} \cup I_{2i})$, there exists an α in $(V_{2i} \cup I_{2i})^*$ such that $a \to \alpha \in P_{2i}$, called a set of vertical productions; S_i is the axiom or start symbol in $(V_{2i} \cup I_{2i})$.

The derivations proceed as follows: First a string $S_{i_1} \ldots S_{i_n} \in I_1^*$ is generated using the horizontal production rules of P_1 in G_1, i.e. $S \overset{*}{\Rightarrow} S_{i_1} \ldots S_{i_n} \in I_1^*$. The vertical derivations proceed as follows:

$$\begin{matrix} S_{i_1} \ldots S_{i_1} \\ \Downarrow \\ \alpha^1_{i_1} \ldots \alpha^1_{i_1} \end{matrix}$$

where $\alpha_{i_j}^1 \in (V_{2i} \cup I_{2i})^+$, $i \in \{1, \ldots, k\}$, $j = 1, \ldots, n$, $|\alpha_{i_1}^1|$ need not be equal to $|\alpha_{i_l}^1|$ $j \neq l$, $l = 1, \ldots, n$ and

$$\begin{array}{l} \alpha_{i_1}^1 \alpha_{i_2}^1 \ldots \alpha_{i_n}^1 \\ \Downarrow \\ \alpha_{i_1}^2 \alpha_{i_2}^2 \ldots \alpha_{i_n}^2 \end{array}$$

where $\alpha_{i_j}^2 \in (V_{2i} \cup I_{2i})^+$, $i \in \{1, \ldots, k\}$, $j = 1, \ldots, n$, and $|\alpha_{i_j}^2|$ need not be equal to $|\alpha_{i_l}^2|$ for $j \neq l$, $l = 1, \ldots, n$.

Proceeding in this way we get a derivation

$$\begin{array}{l} S_{i_1} \ldots S_{i_n} \\ \Downarrow \\ \alpha_{i_1}^1 \ldots \alpha_{i_n}^1 \\ \Downarrow \\ \alpha_{i_1}^2 \ldots \alpha_{i_n}^2 \\ \Downarrow \\ \vdots \\ \Downarrow \\ \alpha_{i_1}^m \ldots \alpha_{i_n}^m \end{array}$$

where $\alpha_{i_j}^m \in (I_{2i})^+$, $j = 1, 2, \ldots, n$, $i \in \{1, \ldots, k\}$ and $|\alpha_{i_j}^m|$ need not be equal to $|\alpha_{i_l}^m|$ for $j \neq l$, $l = 1, \ldots, n$. If $|\alpha_{i_j}^m| = m_j$ and if $m = \max\{m_1, \ldots, m_n\}$, then $\alpha_{i_1}^m \ldots \alpha_{i_n}^m$ is said to be a nonrectangular array over I_{2i} of dimension $m \times n$. Let us denote any nonrectangular $m \times n$ array of the above mentioned type as $\sqcap_{mn}$ and the set of all nonrectangular arrays over I $(\neq \emptyset)$ to be $\lceil I \rceil^{++}$ and $\lceil I \rceil^{**} = \lceil I \rceil^{++} \cup \{\lambda\}$ if λ is the empty array. Clearly $I^{**} \subseteq \lceil I \rceil^{**}$ and $I^{++} \subseteq \lceil I \rceil^{++}$.

So the language generated by G is defined as

$$L(G) = \{\sqcap_{mn} \mid S \overset{*}{\Rightarrow} S_{i_1} \ldots S_{i_n} \overset{*}{\Downarrow} \sqcap_{mn},\ m, n \geq 0, \sqcap_{mn} \in \lceil I \rceil^{**}\}.$$

Remark 1.2. Our earlier models are rectangular arrays [3, 4, 5]. Hence we used the notation M_{rs} to denote a rectangular array of dimension $r \times s$ and I^{++}, the set of all rectangular arrays and $I^{**} = I^{++} \cup \{\lambda\}$ where λ is the empty word.

Example 1.3. Let $G = (G_1, G_2)$ be an (E0L-E0L)A system where

$$G_1 = (\{S_0, S_1, S_2\}, \{S_1, S_2\}, \{S_0 \to S_1 S_0 S_1, S_0 \to S_2, S_1 \to S_1, S_2 \to S_2\}, S_0)$$

and

$$G_2 = G_{21} \cup G_{22},$$

where

$$G_{21} = (\{S_1, X\}, \{X\}, \{S_1 \to X, X \to X\}, S_1),$$

$$G_{22} = (\{S_2, X\}, \{X\}, \{S_2 \to XS_2, S_2 \to XXX, X \to X\}, S_2).$$

Then G generates all $\sqcap_{mn}$ arrays $m \geq 3$, $n \geq 1$ which describe the token T (Fig. 1) of different sizes and proportions.

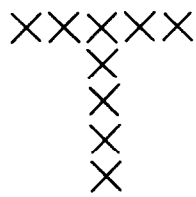

Fig. 1

Example 1.4. Let $G = (G_1, G_2)$ be an (EOL-EOL)AS where $L(G_1) = \{(S_1S_2)^nS_3(S_2S_1)^n \mid n \geq 2\}$, $G_2 = G_{21} \cup G_{22} \cup G_{23}$, $L(G_{21}) = \{a^{2^n}d \mid n \geq 1\}$, $L(G_{22}) = \{(ba)^2c \mid n \geq 1\}$, $L(G_{23}) = \{a^{4^n} \mid n \geq 1\}$. Then G generates the "kolam" pattern of different sizes and proportion where "a" and "d" stand for $\diamond$. and "b" and "c" stand for blank (Fig. 2).

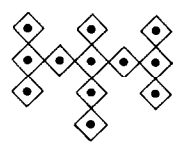

Fig. 2.

Note 1.5. Repetitive patterns occur often in textile designing. Hence several interesting textile patterns and wall-paper designs of [7] can be generated by an (EOL-EOL)AS (Fig. 3, Fig. 4).

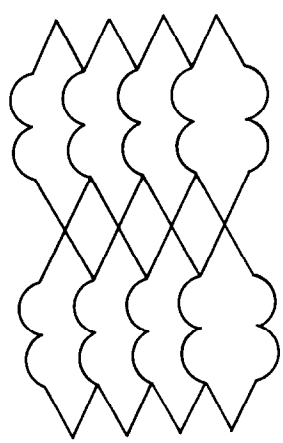

Fig. 3.

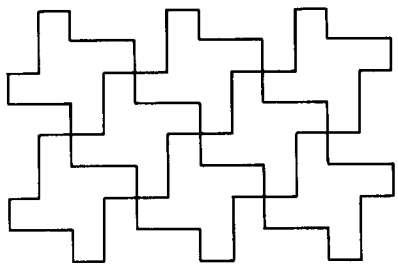

Fig. 4.

Definition 1.6. (i) A (E0L-T0L) array system ((E0L-T0L)AS, (E0L-0L) array system (E0L-0L)AS) is a two-tuple $G=(G_1, G_2)$ where $G_1=(V_1 \cup I_1, I_1, P_1, S_0)$ is a E0L system as in Definition 1.1. $G=\bigcup_{i=1}^{k} G_{2i}$ where $G_{2i}=(I_{2i}, \mathscr{P}_{2i}, w_i)$ is a T0L system (0L system) with $I_2=\bigcup_{i=1}^{k} I_{2i}=I_1$. The derivation of an array is defined exactly as in Definition 1.1.

The set of all arrays generated by G is defined as

$$L(G)=\{\boxed{\ }_{mn} \in \boxed{I_2}^{**},\ m, n \geqslant 0 \mid S_0 \overset{*}{\Rightarrow} S_{i_1} \ldots S_{i_n} \overset{*}{\Downarrow} \boxed{\ }_{mn}\}.$$

(ii) A (T0L-E0L) array system ((T0L-E0L)AS), ((T0L-T0L) array system ((T0L-T0L)AS), (T0L-0L) array system ((T0L-0L)AS) is a two-tuple $G=(G_1, G_2)$ where $G_1=(I_1, \mathscr{P}_1, w)$ is a T0L system with $I_1=\{S_1, \ldots, S_k\}$ (say), the set of intermediates and $G_2=\bigcup_{i=1}^{k} G_{2i}$, each G_{2i} is a E0L (T0L, 0L) system and the derivation is defined exactly as in Definition 1.1. The set of all arrays generated by G is defined as

$$L(G)=\{\boxed{\ }_{mn} \in \boxed{I_2}^{**},\ m, n \geqslant 0 \mid S \overset{*}{\Rightarrow} S_{i_1} \ldots S_{i_n} \overset{*}{\Downarrow} \boxed{\ }_{mn}\}.$$

Similarly (0L-E0L) array system ((0L-E0L)AS), ((0L-T0L) array system ((0L-T0L)AS), (0L-0L) array system ((0L-0L)AS)) and the language generated by it can be defined.

Definition 1.7. A language L is called the $(X-Y)$ array language if there is a $(X-Y)$ array system G such that $L(G)=L$ where $X, Y=$ E0L, T0L, 0L.

Notation. The family of languages generated by X is denoted as $\mathscr{L}(X)$.

2. Machine characterization of (E0L–E0L) array languages

In this section we define an automaton which accepts the family of (E0L-E0L)A languages. A machine model called "restricted pushdown array of counters" automaton accepting the family of E0L languages is introduced by Rozenberg [10]. Extending this idea to two dimensions, we combine the restricted pushdown array of counters automata (RPAC automata) with a finite set of restricted pushdown array of counter automata to obtain a new machine called (Restricted Pushdown Array of Counters-Restricted Pushdown Array of Counters) array automaton which accepts (E0L-E0L) array languages.

First we define the (Pushdown Array of Counters-Pushdown Array of Counters) array automaton and illustrate the machine with an example.

Definition 2.1. A (Pushdown Array of Counters-Pushdown Array of Counters) array automaton ((PAC-PAC)AA) is a twelve-tuple

$$M = \langle K, I, T_1, T_2, T_3, \$, q_0, \delta, \delta', S, F, F' \rangle$$

where

(1) $K = \bar{K} \cup K_1 \cup \cdots \cup K_k$, $K_i \cap K_j = \emptyset$ for $i \neq j$, a finite set of states.

(2) I is a finite set of input symbols.

(3) T_1 is a finite set of initial storage symbols with $\# T_1 = k$, $T_1 = \{s_i \mid S_i \in S, i = 1, \ldots, k, \# S = k\}$ and $S = \bigcup_{i=1}^{k} S_i$ is the set of start states and each storage symbol corresponds to one and only one K_i. T_2 is a finite set of intermediate storage symbols where $T_1 \subsetneq T_2$ and T_3 is the finite set of final storage symbols, where $T_1 \cup I \subsetneq T_3$, $T_3 / I = \bigcup_{i=1}^{k} T_{3_i}$, $T_{3_i} \cap T_{3_j} = \emptyset$ for $i \neq j$, where T_{3_i} is the storage corresponding to K_i for $1 \leq i \leq k$.

(4) $F' = \{q_{d_1}, \ldots, q_{d_k}\}$ is the set of final states and K_i has q_{d_i} as it's final state, for $1 \leq i \leq k$. $\bar{K}$ has a central state q_0 and final state set $F \subset \bar{K}$. $\$ \notin I$ is the end marker.

(5) δ is a (partial) function (called the basic transition function) from $(T_{3_i} \cup \{\lambda\} \cup I) \times K_i$ into finite subsets of $(T_{3_i} \cup \{\lambda\} \cup I) \times K_i$, $i = 1, \ldots, k$ and is also from $(q_{d_i} \times \$)$ into finite subsets of $(S \cup \{q_0\}) \times s_i$ where s_i is in the initial storage T_1 corresponding to K_i, $i = 1, \ldots, k$.

δ' is a (partial) function (called the basic transition function) from $(T_2 \cup \{\lambda\}) \times \bar{K}$ into finite subsets of $(T_2 \cup \{\lambda\}) \times \bar{K}$.

The movement of the automaton is as follows: Place the input array with end markers as shown below:

$$\left| \begin{matrix} \alpha_1 \ldots \alpha_n \\ \$ \ldots \$ \end{matrix} \right| \quad \text{where} \quad \alpha_t = \left| \begin{matrix} a_{1t} \\ \vdots \\ a_{p_t t} \end{matrix} \right|. \tag{1}$$

p_i need not be equal to p_j for $i \neq j$, i.e. $p_1, p_2, \ldots, p_n$ need not be of equal lengths, for $t = 1, \ldots, n$, $a_{j_l} \in I$, $1 \leq j \leq p_t$, $1 \leq l \leq t$, $p_t \geq 1$.

The automaton first reads the first column according to δ and when it reaches \$, it prints out a symbol from T_1 on the storage and goes to the top of the second column and starts reading the second column. Note that the basic transition step of such a machine depends on the pushdown array in a way determined by the bottom symbol of the topmost counter, however moving into the pushdown array is possible only if the two topmost counters store the same number. Similarly it acts on the other columns. If the input array is of type $m \times n$ (where m is $\max\{p_t \mid 1 \leq t \leq n\}$) at the end of δ moves, when the input pointer has read the (m, n)th element, the storage will contain n symbols. The automaton now starts scanning the storage, having it as input. Now it uses δ' moves to scan from left to right. After reading this input it must enter one of the distinguished final states in F. Then the input is

accepted. Otherwise it is rejected. If the automaton encounters a blank in the middle it stops scanning and rejects the input. Hence a (PAC-PAC)A automaton accepts an array $\sqcap_{mn}$ if and only if, when started in one of the initial states in S with $\sqcap_{mn}$ on its input tape and its pushdown array empty, it will in a finite number of steps enter one of the distinguished final states in F with both input tape and pushdown array empty.

A configuration is a five-tuple $(j, y, (i,j), p, x)$ where p is the current state; (i,j) is the position of the input pointer, i.e. the element in the ith row, jth column; $y \in T_1^*$; j is the number of cells from the left end of the pointer of the storage and

$$x \in \bigcup_{i=1}^{k} ((I \cup T_{3_i}) \times N)^* \cup (T_2 \times N)^*.$$

If $b_1 = (j, y, (i,j), p, x)$ is a configuration and if (1) is the input array with

$$x = \langle z_1, v_1\rangle \ldots \langle z_t, v_t\rangle \in (T_{3_j} \times N)^* \text{ and } b_2$$

is any other configuration then we say that $b_1 \vdash b_2$ if one of the following holds: a_1 is the (i,j)th element from I, $p \in K_j$.

(i) [READ INPUT] $i \geq 1$, $p = S_j$ and $b_2 = (j, y, (i+1, j), S_j, \langle z_1, v_1\rangle \ldots \langle z_t, v_t\rangle\langle a_1, 0\rangle$.

(ii) [OVER WRITE] $t \geq 1$, $b_2 = (j, y, (i,j), S_j, \langle z_1, v_1\rangle \ldots \langle z_{t-1}, v_{t-1}\rangle\langle z, v_t + 1\rangle)$ where $(z, S_j) \in \delta(z_t, p)$.

(iii) [POP-UP] $t \geq 2$, $v_{t-1} = v_t$ and $b_2 = (j, y, (i,j), \bar{p}, \langle z_1, v_1\rangle \ldots \langle z_{t-1}, v_{t-1}\rangle)$ where $(\lambda, \bar{p}) \in \delta(z_t, p)$, $\bar{p} \neq S_j$.

(iv) [POP-UP] $t = 1$, $b_2 = (j, y, (i,j), \bar{p}, \lambda)$ where $(\lambda, \bar{p}) \in \delta(z_t, p)$, $\bar{p} \neq S_j$.

(v) [PUSH-DOWN] $t \geq 1$, $p = S_j$ and $b_2 = (j, y, (i,j), S_j, \langle z_1, v_1\rangle \ldots \langle z_{t-1}, v_{t-1}\rangle \langle z_t, v_t\rangle\langle z, v_t\rangle)$ where $(z, S_j) \in \delta(\lambda, S_j)$.

(vi) [PUSH-DOWN] $t = 0$, $p = S_j$ and $b_2 = (j, y, (i,j), S_j, \langle z, 1\rangle$ where $(z, S_j) \in \delta(\lambda, S_j)$.

If \$ is the (i,j)th element and $(z, p') \in \delta(\$, p)$, $p \in F'$, $p' \in S$, then

$$(j, y, (i,j), p, \lambda) \underset{M}{\vdash} (j+1, yz, (1, j+1), p', \lambda)$$

where "$\vdash_M$" means "derives according to M". This is true for all $j \leq n-1$ but if $j = n$, then $p' = q_0$ and if $B = (n, y, (m+1, n), p, \lambda)$, then $B \vdash_M (n, yz, (m+1, n), q_0, \lambda)$, $(z, p') \in \delta(\$, p)$.

Let $B_1 = (n, y, (m+1, n), p, x)$ be a configuration where $y \in T_1^*$, $x \in (T_2 \times N)^*$, $p \in \bar{K}$. We say that B_1 directly derives B_2 in A denoted as $B_1 \vdash_M B_2$ if one of the following holds: If $y = s_1 \ldots s_n \in T_1^*$, $s_i \in T_1$, $i = 1, \ldots, n$, $x = \langle z_1, v_1\rangle \ldots \langle z_t, v_t\rangle \in (T \times N)^*$.

(i) [READ INPUT] $n \geq 1$, $p = q_0$ and

$$B_2 = (n, s_2 \ldots s_n, (m+1, n), q_0, \langle z_1, v_1\rangle \ldots \langle z_t, v_t\rangle\langle s_1, 0\rangle).$$

(ii) [OVER WRITE] $t \geq 1$,

$$B_2 = (n, y, (m+1, n), q_0, \langle z_1, v_1\rangle \ldots \langle z_{t-1}, v_{t-1}\rangle\langle z, v_t + 1\rangle)$$

where $(z, q_0) \in \delta'(z_n, p)$.

(iii) [POP-UP] $t=1$ and $B_2=(n, y, (m+1, n), \bar{p}, \lambda)$ where $(\lambda, \bar{p}) \in \delta'(z_t, p)$, $\bar{p} \neq q_0$.

(iv) [POP-UP] $t \geq 2$, $v_{t-1}=v_t$ and $B_2=(n, y, (m+1, n), \bar{p}, \langle z_1, v_1\rangle \ldots \langle z_{t-1}, v_{t-1}\rangle)$, where $(\lambda, \bar{p}) \in \delta'(z_t, p)$, $\bar{p} \neq q_0$.

(v) [PUSH-DOWN] $t \geq 1$, $p=q_0$ and

$$B_2=(n, y, (m+1, n), q_0, \langle z_1, v_1\rangle \ldots \langle z_t, v_t\rangle\langle z, v_t\rangle)$$

where $(z, q_0) \in \delta'(\lambda, q_0)$.

(vi) [PUSH-DOWN] $t=0$, $p=q_0$ and $B_2=(n, y, (m+1, n), q_0, \langle z, 1\rangle)$ where $(z, q_0) \in \delta'(\lambda, q_0)$.

$\overset{*}{\underset{M}{\vdash}}$ is the reflexive transitive closure of $\underset{M}{\vdash}$.

Let $b_0, \ldots, b_k$ be a sequence of configurations, for $b_0=(1, \lambda, (1, 1), q, \lambda)$ and $b_k=(n, \lambda, (m+1, n), q', \lambda)$ where $q \in S$, $q' \in F$ and $\sqcap_{mn}$ is the input array in $\boxed{I}^{**}$. Then

$$b_0 \underset{M}{\vdash} b_1 \underset{M}{\vdash} b_2 \underset{M}{\vdash} \cdots \underset{M}{\vdash} b_k \quad \text{or} \quad b_0 \overset{+}{\underset{M}{\vdash}} b_k,$$

is called a chain of configurations deriving $\sqcap_{mn}$ in M.

Definition 2.2. The set of all arrays accepted by the (PAC-PAC)A automaton M is defined to be

$$L(M)=\{\sqcap_{mn}, m \geq 0, n \geq 0 \mid (1, \lambda, (1, 1), q, \lambda) \overset{*}{\vdash} (n, y, (m+1, n), q_0, \lambda) \overset{*}{\vdash} (n, \lambda, (m+1, n), q', \lambda) \text{ with } q \in S, q' \in F, y \in T_1^*\}.$$

Definition 2.3. Let $M=\langle K, I, T_1, T_2, T_3, \$, q, \delta, \delta', S, F, F'\rangle$ be a (PAC-PAC)A automaton.

(1) Let e, e' be any two positive integers. M is called (e, e')-restricted if and only if whenever the input array in $\boxed{I}^{**}$ is of dimension $m \times n$ and $b_{0j}, b_{1j}, \ldots, b_{e_j j}$ and $b_0, b_1, \ldots, b_{e'}$ are $(e+e')$-sets of configurations for $j=1, \ldots, k$, $\#S=k$ such that

$$b_{0j} \vdash b_{1j} \vdash \cdots \vdash b_{e_j j}, 1 \leq j \leq k \quad \text{and} \quad b_0 \vdash b_1 \vdash \cdots \vdash b_{e'},$$

then $b_{ij}=(j, y, (1, j), S_j, x)$ for $i \in \{1, \ldots, e_j\}$, $y \in T_1^*$, $x \in ((T_{3j} \cup I) \times N)^*$, $1 \leq j \leq k$ and $e=\max\{e_j \mid 1 \leq j \leq k\}$ and $b_i=(n, y, (m+1, n), q_0, x)$ for some $i \in \{1, \ldots, e'\}$, $y \in T_1^*$, $x \in (T_2 \times N)^*$.

(2) M is called a (restricted PAC-restricted PAC) array automaton abbreviated as (RPAC-RPAC)AA if it is (e, e')-restricted for some $e \geq 1$, $e' \geq 1$.

(3) M is called λ-free if $(\lambda, S_j) \notin \text{Dom } \delta$, $j=1, \ldots, k$ where Dom δ is the domain of δ, $\#S=k$ and $(\lambda, q_0) \notin \text{Dom } \delta'$.

(4) M is called simple if

$$I=\{a \in T_3 \mid (a, q) \in \text{Dom } \delta \text{ for some } q \in (K/\bar{K})\}$$

and

$$T_1 = \{s \in T_2 \mid (s, q) \in \text{Dom } \delta' \text{ for some } q \in \bar{K}\}.$$

Definition 2.4. A language L is called a (restricted, simple, λ-free) (PAC-PAC)A language if there exists a (restricted, simple, λ-free) (PAC-PAC)A automaton which accepts L. (RPAC-RPAC)A languages are the set of (restricted PAC-restricted PAC) array languages.

Let us state the following lemmas without proofs as they follow from the definitions.

Lemma 2.5. *Let* $M = \langle K, I, T_1, T_2, T_3, \$, q_0, \delta, \delta', S, F, F' \rangle$ *be a (PAC-PAC)A automaton.*

(1) (i) *If* b_j *is any configuration such that* $b_j = (j, y, (1, j), p, \lambda)$ *for* $p \in K_j - \{S_j\}$ *for any* $y \in T_1^*$, *for any input array* $\boxed{\ }_{mn} \in \boxed{I}^{**}$, *then there does not exist a configuration* $\bar{b}_j$ *such that* $b_j \vdash \bar{b}_j$, $j = 1, \ldots, k$.

(ii) *If* b *is any configuration such that* $b = (n, y, (m+1, n), p, \lambda)$ *for* $p \in \bar{K} - \{q_0\}$ *and for any* $\boxed{\ }_{mn} \in \boxed{I}^{**}$, input array, then there does not exist a configuration $\bar{b}$ such that $b \vdash \bar{b}$.

(2) (i) *For no configuration* $b_j, j = 1, \ldots, k$, $b_j = (j, y, (i, j), p, x)$ *for* $p \in K_j - \{S_j\}$ *for any* $y \in T_1^*$, *for any input array* $\boxed{\ }_{mn} \in \boxed{I}^{**}$, $b_j \vdash b_j$, $x \in ((T_{3j} \cup I) \times N)^*$.

(ii) *For no configuration* b *such that* $\boxed{\ }_{mn} \in \boxed{I}^{**}$, $b = (n, y, (m+1, n), p, x)$, $y \in T_1^*$, $x \in (T_2 \times N)^*$, $b \vdash b$.

(3) *If* $\bar{M} = \langle K, I, T_1, T_2, T_3, \$, q_0, \delta, \delta', S, F - \{q_0\}, F' - \{S_1, \ldots, S_k\} \rangle$, $\# S = k$ *is a (PAC-PAC)A automaton, then* $L(M) = L(\bar{M})$.

Lemma 2.6. *Let* $M = \langle K, I, T_1, T_2, T_3, \$, q_0, \delta, \delta', S, F, F' \rangle$ *be a (PAC-PAC)A automaton. For any* $j \in \{1, \ldots, k\}$. $\# S = k$, $t_j \geq 1$, $\alpha_{1_j}, \ldots, \alpha_{t_j}, \beta_{1_j}, \ldots, \beta_{t_j}, \beta_j \in I^*$, $y_{1_j}, \ldots, y_{t_j} \in ((T_{3_j} \cup I) \times N)^*$ *for* $|\alpha_{i_j}| = l_{i_j}$, $1 \leq i \leq t$, $l_{i_j} \geq 1$, $\alpha_{i_j}\beta_{i_j}$ *being the input at the jth column. If*

$$(j, y, (1, j), S_j, \lambda) \overset{+}{\vdash} (j, y, (l_{1_j}, j), S_j, y_{1_j})$$

$$(j, y, (1, j), S_j, \lambda) \overset{+}{\vdash} (j, y, (l_{2_j}, j), S_j, y_{2_j})$$

$$\vdots$$

$$(j, y, (1, j), S_j, \lambda) \overset{+}{\vdash} (j, y, (l_{t_j}, j), S_j, y_{t_j})$$

then for the input $\alpha_{1_j} \ldots \alpha_{t_j}\beta_j$, $j = 1, 2, \ldots, k$,

$$(j, y, (1, j), S_j, \lambda) \overset{+}{\vdash} \left(j, y, \left(\sum_{i=1}^{t} l_{i_j}, j\right), S_j, y_{1_j} \ldots y_{t_j}\right).$$

If $\boxed{\ }_{mn} \in \boxed{I}^{**}$ *is the input array, and if* $t \geq 1$, $\alpha_1, \ldots, \alpha_t, \beta_1, \ldots, \beta_t, \beta \in T_1^*$,

$y_1, \ldots, y_t \in (T_2 \times N)^*$ *and if*

$$(n, \alpha_1\beta_1, (m+1, n), q_0, \lambda) \overset{+}{\vdash} (n, \beta_1, (m+1, n), q_0, y_1)$$

$$(n, \alpha_2\beta_2, (m+1, n), q_0, \lambda) \overset{+}{\vdash} (n, \beta_2, (m+1, n), q_0, y_2)$$

$$\vdots$$

$$(n, \alpha_t\beta_t, (m+1, n), q_0, \lambda) \overset{+}{\vdash} (n, \beta_t, (m+1, n), q_0, y_t)$$

then

$$(n, \alpha_1 \ldots \alpha_t\beta, (m+1, n), q_0, \lambda) \overset{+}{\vdash} (n, \beta, (m+1, n), q_0, y_1, \ldots, y_t$$

with $|\alpha_1 \ldots \alpha_t\beta| \leq n$.

Next let us characterize (RPAC-RPAC)A languages.

Theorem 2.7. *There exists an algorithm which for every* (E0L-E0L)A system G *constructs an* (*RPAC-RPAC*)*A automaton* M *such that* $L(G) = L(M)$. *Moreover, if* G *is propagating, then* M *is* λ*-free.*

Proof. Let $G = (G_1, G_2)$ where $G_1 = (V_1 \cup I_1, I_1, P_1, S_0)$ and $G_2 = \bigcup_{i=1}^{k} g_{2i}$ where $G_{2i} = (V_{2i} \cup I_{2i}, I_{2i}, P_{2i}, S_i)$ with $\# I_1 = k$ and $I_{2i} \cap I_{2j}$ need not be empty for $i \neq j$, $I_2 = \bigcup_{i=1}^{k} I_{2i}$, be an (E0L-E0L) array system, $V_{2i} \cap V_{2j} = \emptyset$ for $i \neq j$. The productions in P_1 are of the following form:

$$\left.\begin{array}{l} A_1 \to B_{1n_1} \ldots B_{11} \\ A_2 \to B_{2n_2} \ldots B_{21} \\ \quad \vdots \\ A_m \to B_{mn_m} \ldots B_{m1} \\ A_{m+1} \to \lambda \\ A_{m+2} \to \lambda \\ \quad \vdots \\ A_{m+r} \to \lambda \end{array}\right\} \tag{1}$$

For some m, $r \in N$, $m + r \geq 0$, $n_i \geq 1$ for $i \in \{1, \ldots, m\}$ and $A_j \in V_1 \cup I_1$, $j \in \{1, \ldots, m+r\}$. The productions in P_{2i} of G_{2i} are

$$a_{1_i} \to b_{1n_{1_i}} \ldots b_{11}$$

$$a_{2_i} \to b_{2n_{2_i}} \ldots b_{21}$$

$$\vdots$$

$$a_{m_i} \to b_{m_i n_{m_i}} \ldots b_{m_i 1}$$

$$a_{m_i+1} \to \lambda, \ldots, a_{m_i+r_i} \to \lambda$$

for some $m_i, r_i \in N$, $m_i + r_i \geq 0$, $n_j \geq 1$ for $j \in \{1, \ldots, m_i\}$ and $a_t \in (V_{2i} \cup I_{2i})$, $t \in \{1, \ldots, m_i + r_i\}$. This is true for all $1 \leq i \leq k$.

The effective construction of $M = \langle K, I, T_1, T_2, T_3, \$, q_0, \delta, \delta', S, F, F' \rangle$ is as follows:

(I) $K = \bar{K} \cup K_1 \cup \cdots \cup K_k$, $K_i \cap K_j = \emptyset$ for $i \neq j$ where

$$\bar{K} = \{q_0\} \cup \{[t, j]: 1 \leq t \leq m, n_t \geq 2, 2 \leq j \leq n_t\} \cup \{q_d\}.$$

$$K_i = \{S_i\} \cup \{[t, j]_i: 1 \leq t \leq m_i, n_t \geq 2, 2 \leq j \leq n_t\} \cup \{q_{d_i}\},$$

for $1 \leq i \leq k$.

(II) $F' = \{q_{d_1}, \ldots, q_{d_k}\}$.

(III) $F = \{q_d\}$.

(IV) $S = I_1 = \{S_1, \ldots, S_k\}$.

(V) $I_2 = I$, the set of input symbols.

(VI) $T_1 = \{s_i \mid S_i \in S, i = 1, \ldots, k, \# S = k\}$, $S = \bigcup_{i=1}^{k} S_i$ is the set of start states.

(VII) $T = T_1 \cup V_1$, $T_3 = \bigcup_{i=1}^{k} V_{2i} \cup I_2 = \bigcup_{i=1}^{k} V_{2i} \cup I$, $V_{2i} \cap V_{2j} = \emptyset$ for $i \neq j$, $T_{3_i} = V_{2_i}$, $1 \leq i \leq k$.

(VIII) q_0 is the central state, $\$ \notin I$ is the end-marker.

(IX) The basic transition function δ is defined as follows: $\delta = \delta_i$ for the E0L system G_{2i} and δ_i is defined as follows: Le γ be a function from T_3 to $T_1 \cup \{\bigcup_{i=1}^{k} (V_{2i} \mid \{S_i\})\} \cup \{I\}$ defined as $\gamma(A) = a$ or A according as $A \in S$ or $A \in T_3/S$, for $1 \leq i \leq k$.

(1) If $n_t = 1$, for some $t \in \{1, \ldots, m_i\}$, then

$$(\gamma(a_t), S_i) \in \delta_i(\gamma(b_{tn_t}), S_i).$$

(2) If $n_t > 1$ for some $t \in \{1, \ldots, m_i\}$ then

$$(\lambda, [t, 2]_i) \in \delta_i(\gamma(b_{t1}), S_i).$$

(3) If $t \in \{1, \ldots, m_i\}$ and $j < n_t$, then

$$(\lambda, [t, j+1]_i) \in \delta_i(\gamma(b_{tj}), [t, j]_i).$$

(4) For every $t \in \{1, \ldots, m_i\}$,

$$(\gamma(a_t), S_i) \in \delta_i(\gamma(b_{tn_t}), [t, n_t]_i).$$

(5) For every $t \in \{1, \ldots, r_i\}$, $(\gamma(a_{m_i+t}), S_i) \in \delta_i(\lambda, S_i)$.

(6) $(\lambda, q_{d_i}) \in \delta_i(\gamma(S_i), S_i)$.

(7) For every $z_i, y_i \in (V_{2i} \cup I) \cup \{\lambda\}$, $q_i, \bar{q}_i \in K_i$,

$$(\gamma(z_i), \bar{q}_i) \in \delta_i(\gamma(y_i), q_i)$$

only if this was implied by (1) to (6) of this construction.

(8) Also $\delta(\$, q_{d_i}) = (s_i, S_j)$, $j = 1, \ldots, k$, $i = 1, \ldots, k$. Hence the construction δ is over. For δ' we have q_0 as the central, q_d as the final state, T_2 as the storage symbols and T_1 as the input. Let γ' be a function from $V_1 \cup I_1$ to $V_1 \cup T_1$ defined as follows:

$\gamma'(A) = a$ or A according as $A \in I_1$ or $A \in V_1$. The construction of δ' is exactly similar to that of the construction of δ_i. G is propagating if M is λ-free.

The construction is complete. We can prove that $L(G) = L(M)$ using the following lemma. □

As the proof of the lemma and the proof of $L(G) = L(M)$ are similar to those in [10] we omit the proofs and just state the lemma as follows.

Lemma 2.8. *Let* $\boxed{\ }_{mn}$ *be an input array over* $\boxed{I}^{**}$. *Let M be any (RPAC-RPAC)A automaton constructed as in Theorem* 2.7 *from a* (E0L-E0L)*A system* $G = (G_1, G_2)$.

(i) *For* $1 \leq i \leq k$, *let*

$$\Gamma_i = (j, y, (1, j), S_i, \lambda), \quad \textit{and}$$

$$\Gamma_i' = (j, y, (f, j), S_i, \langle z_1, v_1\rangle \ldots \langle z_{n_i}, v_{n_i}\rangle)$$

where $a_1 \ldots a_{m_i} \in I^*$ *being the input at the jth column,* $\langle z_t, v_t\rangle \in (\{V_{2i} \cup I\} \times N)$ *for* $t = \{1, \ldots, n_i\}$, *be two configurations such that,* $m_i \geq 0$, $f \leq m_i$, $n_i \geq 1$, $v_1, \ldots, v_{n_i} \geq 1$. *Then* $\Gamma_i \overset{*}{\underset{M}{\vdash}} \Gamma_i'$ *if and only if there exists* $u_1, \ldots, u_{n_i} \in (V_{2i} \cup I)^*$ *such that*

$$z_1 \overset{v_1}{\underset{G}{\Rightarrow}} u_1, \ldots, z_{n_i} \overset{v_{n_i}}{\underset{G}{\Rightarrow}} u_{n_i}$$

and $a_1 \ldots a_{f-1} = u_1 \ldots u_{n_i}$ *where* $G = (G_1, G_2)$ *and M is defined as in Theorem* 2.7.

(ii) *Let*

$$\Gamma = (n, y, (m+1, n), q_0, \lambda), \bar{\Gamma} = (n, y', (m+1, n), q_0, \langle z_1, v_1\rangle \ldots \langle z_g, v_g\rangle)$$

where $y = a_1 \ldots a_n \in T_1^*$, $y' = a_f \ldots a_n$, $g \geq 1$, $v_1, \ldots, v_g \geq 1$, $n \geq 0$, $f \leq n$ *be two configurations. Then* $\Gamma \overset{+}{\underset{M}{\vdash}} \bar{\Gamma}$ *if and only if there exists* $u_1, \ldots, u_g$ *in* $(T_1 \cup V_1)^*$ *such that*

$$z_1 \overset{v_1}{\underset{G}{\Rightarrow}} u_1, \ldots, z_g \overset{v_g}{\underset{G}{\Rightarrow}} u_g$$

and $a_1 \ldots a_{f-1} = u_1 \ldots u_g$ *where* $G = (G_1, G_2)$ *and M are defined as in Theorem* 2.7.

Theorem 2.9. *There exists an algorithm which for every (RPAC-RPAC)A automaton M constructs an (E0L-E0L)A system* $G = (G_1, G_2)$ *such that* $L(M) = L(G)$. *Moreover, if M is* λ*-free, then G is propagating.*

Proof. Only the outline of the proof is given as it is similar to the proof of Theorem 2.7. Let $M = \langle K, I, T_1, T_2, T_3, \$, q_0, \delta, \delta', S, F, F'\rangle$ where $K = \bar{K} \cup K_1 \cup K_2 \cup \cdots \cup K_k$, $\# T_1 = k$ be a (RPAC-RPAC)A automaton. Then $G = (G_1, G_2)$, an (E0L-E0L)A system is constructed such that $L(G) = L(M)$ as follows: Before taking up the effective construction let us define the following.

Let $\sqcap_{mn}$ be an input array of dimension $m \times n$. If $\alpha_{j_i} = a_{1i} \ldots a_{t_i i}$, $a_{s_i i} \in (T_{3_i} \cup I)$, $b_i \in T_{3_i} \cup I)$, for any $j \in \{1, \ldots, n\}$, α_{j_i} is called a b_i-word if

$$(j, y, (t_i+1, j), S_i, \langle a_{1i}, 0\rangle \ldots \langle a_{t_i i}, 0\rangle)$$
$$\vdash (j, y, (t_i+1, j), q_{1i}, \langle a_{1i}, 0\rangle \ldots \langle a_{t_i-1,i}, 0\rangle)$$
$$\vdash \cdots \vdash (j, y, (t_i+1, j), q_{t_i-1,i}, \langle a_{1i}, 0\rangle)$$
$$\vdash (j, y, (t_i+1, j), S_i, \langle b_i, 1\rangle)$$

for some $q_{1i}, \ldots, q_{t_i-1,i}$ in K_i such that for every $p \in \{1, \ldots, t_i - 1\}$, $q_{pi} \neq S_i$. This is true for all i, $1 \leqslant i \leqslant k$. Also α_{j_i} is called i-initial (in M) if

$$(j, y, (t_i+1, j), S_i, \langle a_{1i}, 0\rangle \ldots \langle a_{t_i i}, 0\rangle)$$
$$\vdash (j, y, (t_i+i, j), q_{1i}, \langle a_{1i}, 0\rangle \ldots \langle a_{t_i-1}, 0\rangle)$$
$$\vdash \cdots \vdash (j, y, (t_i+1, j), q_{t_i-1,i}, \langle a_{1i}, 0\rangle)$$
$$\vdash (j, y, (t_i+1, j), q_{t_i i}, \lambda)$$

for some $q_{1i}, \ldots, q_{t_i i} \in K_i$ such that $q_{t_i i} \in F'$ and for every $p \in \{1, \ldots, t_i\}$, $q_{pi} \neq S_i$. This is true for all $1 \leqslant i \leqslant k$, $\neq T_1 = k$.

Let $Y = A_1 \ldots A_t$ for some $t \geqslant 1$, $A_j \in T_2$, $1 \leqslant j \leqslant t$ and $B \in T_2$. Then Y is called a B-word if

$$(n, \lambda, (m+1, n), q_0, \langle A_1, 0\rangle \ldots \langle A_t, 0\rangle)$$
$$\vdash (n, \lambda, (m+1, n), q_1, \langle A_1, 0\rangle \ldots \langle A_{t-1}, 0\rangle)$$
$$\vdash \ldots \vdash (n, \lambda, (m+1, n), q_{t-1}, \langle A_1, 0\rangle)$$
$$\vdash (n, \lambda, (m+1, n), q_0, \langle B, 1\rangle)$$

for some $q_1, \ldots, q_{t-1} \in \bar{K}$ such that for every $j \in \{1, \ldots, t-1\}$, $q_j \neq q_0$.

Y is called initial in M if

$$(n, \lambda, (m+1, n), q_0, \langle A_1, 0\rangle \ldots \langle A_t, 0\rangle)$$
$$\vdash (n, \lambda, (m+1, n), q_1, \langle A_1, 0\rangle \ldots \langle A_{t-1}, 0\rangle)$$
$$\vdash \ldots \vdash (n, \lambda, (m+1, n), q_{t-1}, \langle A_1, 0\rangle)$$
$$\vdash (n, \lambda, (m+1, n), q_t, \lambda)$$

for some $q_1, \ldots, q_t \in \bar{K}$, such that $q_t \in F$ and for every $j \in \{1, \ldots, t\}$, $q_j \neq q_0$.

Note that as M is an (RPAC-RPAC)A automaton, there exist positive integers e (e') such that for every i-initial (initial), an b_i-word (B-word) exists not larger than e (e'), for some $b_i \in (T_{3_i} \cup I)$ ($B \in T_2$). Clearly the above properties are decidable.

Now let us take up the construction of $G = (G_1, G_2)$.

(I) The construction of G_2. Let τ be a function from $T_3/\{S\}$ to T_3 defined as follows: $\tau(s_i) = S_i$ if $s_i \in T_1$ and $\tau(s_i) = s_i$ if $s_i \in T_3/\{S\}$. Let $G_{2i} = (V_{2i} \cup I_{2i} \cup \{S^i, f^i\}$,

I_{2i}, P_{2i}, S^i), $I_{2i} \in I_2$ be an E0L system where $f^i, S^i \notin V_{2i}$ and S^i is the start symbol, $V_{2i} = T_{3i}$, $T_3 = \bigcup_{i=1}^{k} T_{3i} \cup I_2$ and P_{2i} consists of the following productions.

(1) If $(a_i, S_i) \in \delta_i(\lambda, S_i)$ for some $a_i \in T_{3i} \cup I_{2i}$, then $\tau(a_i) \to \lambda \in P_{2i}$.

(2) If $(b_i, S_i) \in \delta_i(z_i, S_i)$ for some $b_i, z_i \in T_{3i} \cup I_{2i}$, then $\tau(b_i) \to \tau(z_i) \in P_{2i}$.

(3) If $a_{1i} \dots a_{t_i i}$ is an i-initial word then $S^i \to \tau(a_{1i} \dots a_{t_i i})$ is in P_{2i}.

(4) If $a_1^i \dots a_t^i$ is a b^i-word for some $b^i \in V_{2i} \cup I_{2i}$, then $\tau(b^i) \to \tau(a_1^i \dots a_t^i) \in P_{2i}$.

(5) For every letter $x \in V_{2i} \cup I_{2i} \cup \{S^i, f^i\}$, if no production for x was specified in (1) to (4) then $\tau(x) \to f^i \in P_{2i}$.

This is true for all i, $1 \leq i \leq k$.

$$I_{2i} = \{a: a \in I_2, A \to \alpha \in P_{2i}, A \in V_{2i}, a \in \text{alph}(\alpha)\}$$
$$\cup \{a: a \in I_2, a \to \alpha \in P_{2i}, \text{either } a \text{ occurs in } \alpha$$
$$\text{or on the L.H.S. of } a \to \alpha \text{ in } P_{2i}\}.$$

(II) The construction of $G_1 = (V_1 \cup I_1, I_1, P_1, S_0')$. Let τ' be a function from $T_2 \to I_1 \cup \{T_2/T_1\}$ where $I_1 = \{S_i \mid s_i \in T_1\}$, $\tau'(s_i) = S_i$ or s_i according as $s_i \in T_1$ or $s_i \in (T_2/T_1)$. In G_1, $V_1 \cup I_1 = (T_2/T_1) \cup I_1 \cup \{S_0', f\}$. P_1 is obtained as follows where $S_0', f \notin (T_2/T_1) \cup I_1$, S_0' is the start symbol.

(1) If $(A, g_0) \in \delta'(\lambda, q_0)$ for some $A \in T_2$, then $\tau'(A) \to \lambda \in P_1$.

(2) If $(B, q_0) \in \delta'(z, q_0)$ for some B, $z \in T_2$, then $\tau'(B) \to \tau'(z) \in P_1$.

(3) If $A_1 \dots A_t$ is a B-word for some $B \in T_2$, then $\tau'(B) \to \tau'(A_1 \dots A_t) \in P_1$.

(4) If $A_1 \dots A_t$ is an initial word then $S_0' \to \tau'(A_1 \dots A_t) \in P_1$.

(5) For every letter $X \in T_2 \cup \{S_0', f\}$ if no production for X was specified through (1) to (4), then $\tau'(X) \to f \in P_1$.

The construction is complete. Simulation of $L(G) = L(M)$ is as follows: For an (E0L-E0L)A system construct an (RPAC-RPAC)A machine M' and the rest is analogous to the proof of Theorem 2.7. If M is λ-free, then G is propagating. □

Combining Theorem 2.1 and 2.2 we state the major result of this section.

Theorem 2.10. *A language is a* (λ-*free*) (*RPAC-RPAC*)*A language if and only if it is a* (*propagating*) (E0L-E0L)*A language.*

3. Closure properties

In this section we introduce three new operators and one homomorphism which are row cross doubling, column cross doubling, cyclic permutation and column homomorphism h. Let us investigate the closure of the family of (E0L-E0L)AL ((E0L-T0L)AL, (E0L-0L)AL, (T0L-E0L)AL, (T0L-T0L)AL, (T0L-0L)AL, (0L-E0L)AL, (0L-T0L)AL, (0L-0L)AL) under these new operators and under the column homomorphism. Let us also study the closure under the matrix language operators of [13] like column catenation, column +, column * and reflection about the right most vertical of the families of the two dimensional languages introduced in this paper.

Definition 3.1. If $\sqcap_{mn} \in \lceil I \rceil^{**}$ is any $m \times n$ array where $I = \{a_1, a_2, \ldots, a_g\}$ then the cyclic permutation on $\sqcap_{mn}$ is denoted as $\sqcap_c$ and defined to be the array obtained by replacing every a_i by a_{i+1}, $1 \leq i < g$ and a_g replaced by a_1 in each column.

Note 3.2. If $g = 2$, $\sqcap_c$ is the conjugate of $\sqcap$ and denoted as $\sqcap^c$.

Lemma 3.3. *The class of* (E0L-E0L)*AL* ((E0L-T0L)*AL*, (E0L-0L)*AL* (T0L-E0L)*AL* (T0L-T0L)*AL*, (T0L-0L)*AL*, (0L-E0L)*AL*, (0L-T0L)*AL*, (0L-0L)*AL*) *is closed under cyclic permutation.*

Proof. Is omitted as it is simple. □

Definition 3.4. If $X = [\alpha_1 \ldots \alpha_n]$ is an $m \times n$ array in $\lceil I \rceil^{**}$ (say), where

$$\alpha_i = \begin{vmatrix} a_{1i} \\ a_{2i} \\ \vdots \\ a_{m_i i} \end{vmatrix} \quad 1 \leq i \leq n,\ m = \max\{m_i \mid 1 \leq i \leq n\},$$

then column cross doubling of X is

$$X_C X = [\alpha_1 \alpha_1 \ldots \alpha_n \alpha_n]$$

and row cross doubling of X is

$$X_R X = \alpha_1^R \ldots \alpha_n^R$$

where

$$\alpha_i^R = \begin{vmatrix} a_{1i} \\ a_{1i} \\ \vdots \\ a_{m_i i} \\ a_{m_i i} \end{vmatrix} \quad 1 \leq i \leq n.$$

If L is any (E0L-E0L)AL ((E0L-T0L)AL, (E0L-0L)AL, (T0L-E0L)AL, (T0L-T0L)AL, (T0L-0L)AL, (0L-E0L)AL, (0L-T0L)AL, (0L-0L)AL), then the language obtained by performing row cross doubling (column cross doubling) is denoted by L_{rcd} (L_{ccd}) where

$$L_{rcd} = \{X_R X \mid X \in L\}, \qquad L_{ccd} = \{X_C X \mid X \in L\}.$$

Lemma 3.5. (i) *The family of* (E0L-E0L)*AL* ((E0L-T0L)*AL*, (E0L-0L)*AL*) *is closed under column cross doubling.*

(ii) *The family of* (E0L-E0L)*AL* ((T0L-E0L)*AL*, (0L-E0L)*AL*)) *is closed under row cross doubling.*

(iii) *The family of* (T0L-T0L)*AL* ((T0L-0L)*AL*, (0L-T0L)*AL*, (0L-0L)*AL*) *is not closed under column cross doubling and row cross doubling.*

Proof. Follows from the definitions. □

Note 3.6. For any $\boxed{\ }_{mn} \in \boxed{I}^{**}$, $\#I \neq \emptyset$, row k-cross doubling (column k-cross doubling) is defined as follows: *If* $\boxed{\ }_{mn} = [\alpha_1 \ldots \alpha_n]$ where

$$\alpha_i = \begin{vmatrix} a_{1i} \\ \vdots \\ a_{m_i i} \end{vmatrix} \quad 1 \leq i \leq n,\ a_{ji} \in I,\ 1 \leq j \leq m,$$

$m = \max\{m_i \mid 1 \leq i \leq n\}$, row k-cross doubling of $\boxed{\ }$ is denoted as $\boxed{\ }_{R_k}\boxed{\ }$ defined as

$$\boxed{\ }_{R_k}\boxed{\ } = [(\alpha_1)_k \ldots (\alpha_n)_k]$$

where

$$(\alpha_i)_k = \begin{vmatrix} (a_{1i})_k \\ \vdots \\ (a_{m_i i})_k \end{vmatrix} \text{ where } (a_{ji})_k = \left.\begin{vmatrix} a_{ji} \\ a_{ji} \\ \vdots \\ a_{ji} \end{vmatrix}\right\} k \text{ times}$$

for $1 \leq j \leq m_i$, $1 \leq i \leq n$, and column k-cross doubling of $\boxed{\ }$ is denoted as $\boxed{\ }_{C_k}\boxed{\ }$ defined as

$$\boxed{\ }_{C_k}\boxed{\ } = [(\alpha_1)^k \ldots (\alpha_n)^k]$$

where

$$(\alpha_i)^k = \underbrace{\alpha_i \alpha_i \ldots \alpha_i}_{k \text{ times}} \quad \text{for } 1 \leq i \leq n,$$

i.e. L is any (EOL-EOL)AL ((EOL-TOL)AL, (EOL-0L)AL, (TOL-EOL)AL, (TOL-TOL)AL, (TOL-0L)AL (0L-EOL)AL, (0L-TOL)AL, (0L-0L)AL), then the language obtained by performing row k-cross doubling (column k-cross doubling) on L is denoted by $L_{rk\text{-}cd}$ ($L_{ck\text{-}cd}$) and defined as

$$L_{rk\text{-}cd} = \{\boxed{\ }_{R_k}\boxed{\ } \mid \boxed{\ } \in L\} \qquad L_{ck\text{-}cd} = \{\boxed{\ }_{C_k}\boxed{\ } \mid \boxed{\ } \in L\}.$$

Let us now introduce the idea of generalized column homomorphism and array homomorphism.

Definition 3.7. (i) A mapping h from $\boxed{I}^{**}$ to $\boxed{I'}^{**}$ is called a homomorphism h provided

$$h(a) = \begin{vmatrix} b_1 \\ \vdots \\ b_t \end{vmatrix}$$

where $a \in I$, $b_i \in I'$, $1 \leqslant i \leqslant t$, t being the same for all $a \in I$. Then $h(a \cdot b) = h(a) \cdot h(b)$; for any $M \subset \boxed{I}^{**}$, $h(M) = \{h(\boxed{\ }) \mid \boxed{\ } \in M,\ M \subset \boxed{I}^{**}\}$.

(ii) A mapping H from $\boxed{I}^{**}$ $\boxed{I'}^{**}$ is called an array homomorphism provided

$$H(a) = \begin{vmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{vmatrix}, \quad a_{ij} \in I',\ 1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n,$$

m, n being the same for all $a \in I$. The definition is well-defined as each symbol is replaced by a rectangular array. Also

$$H(a \cdot b) = H(a) \cdot H(b) \quad \text{and} \quad H\begin{pmatrix} a \\ \vdots \\ b \end{pmatrix} = \begin{matrix} H(a) \\ \vdots \\ H(b) \end{matrix}$$

Lemma 3.8. (i) *The family of* (E0L-E0L)*AL* ((T0L-E0L)*AL*, (0L-E0L)*AL*) *is closed under column homomorphism h.*

(ii) *The family of* (*E0L-E0L*)*AL is closed under array homomorphism H.*

Remark 3.9. We observe that the families of (E0L-E0L)AL ((E0L-T0L)AL, (E0L-0L)AL) are closed under column catenation, column +, column * of [13], whereas the remaining six families are not closed under the above mentioned operations. All the nine families are closed under the picture language operator-reflection about the rightmost vertical of [13].

References

[1] G.T. Herman and G. Rozenberg, *Developmental Systems and Languages* (North-Holland, Amsterdam 1975).

[2] K. Krithivasan and N. Nirmal, 0L and T0L array languages, *J. Indian. Inst. Sci.* **62** (A) (1980) 101-110.

[3] N. Nirmal and K. Krithivasan, E0L and ET0L array languages, *Proc. Indian. Acad. Sci. Math. Sci.* **90** (1981) 167-180.

[4] N. Nirmal and K. Krithivasan, Table matrix L system, *Internat. J. Comput. Math.* (1982) 247-265.

[5] N. Nirmal and K. Krithivasan, Context dependent table matrix L systems, *Inform. Sci.* **36** (1985) 249-265.

[6] N. Nirmal, Normal forms of extended table matrix L systems, *Inform. Sci.* **39** (1986) 153-173.

[7] N. Nirmal, R. Rama and C. Sri Hari Nagore, Machine characterization of E0L-regular matrix system, *Internat. J. Comput. Math.* **21** (1987) 245-276.

[8] N. Nirmal, R. Rama and C. Sri Hari Nagore, Index on ET0L array systems, *Internat. J. Comput. Math.*, to appear.

[9] A. Rosenfeld, *Picture Languages* (Academic Press, New York, 1979).

[10] G. Rozenberg, On a family of acceptors for some classes of developmental languages, *Internat. J. Comput. Math.* Section A (1974) 199-228.

[11] G. Rozenberg and A. Salomaa, *The Mathematical Theory of* L *Systems* (Academic Press, New York, 1980).

[12] A. Salomaa, *Formal Languages* (Academic Press, New York, 1973).

[13] G. Siromoney, R. Siromoney and K. Krithivasan, Abstract family of matrices and picture languages, *Comput. Graph. Image Process.* (1972) 284-307.